CLAIMS

1. A method of operating a computing system supporting multiple processes, said method including the steps of:

providing a set of monitors for controlling access to resources of the computer system, whereby a process has to enter a monitor in order to access a corresponding resource, and has to join an entry queue for the monitor if the resource is currently owned by another process;

responsive to a predetermined condition, examining processes queued on monitors to determine whether there is a deadlock situation; and

if such a deadlock is found, returning information about the identity of the processes and monitors involved in the deadlock.

2. The method of claim 1, wherein said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process.

3. The method of claim 2, wherein the processes involved in said examining step are determined iteratively based on those processes queued on the requested monitor, and those processes queued on monitors owned by processes already determined to be involved in said examining step.

4. The method of claim 2, wherein the system maintains a global table of contended monitors, a monitor being

contended if there is a process in its entry queue, the table identifying processes which own or are queued on contended monitors.

5. The method of claim 1, wherein said predetermined condition comprises the detection that a set of processes have not progressed over a certain interval.

6. The method of claim 5, wherein said detection is performed by a process that periodically examines the program counter of other processes on the system.

7. The method of claim 1, wherein an error is returned if deadlock is found, said error including said information about the identity of the processes and monitors involved in the deadlock.

8. The method of claim 1, wherein an exception is returned if deadlock is found, said exception including said information about the identity of the processes and monitors involved in the deadlock.

9. The method of claim 8, wherein said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process, and said exception is returned to the requesting process.

10. The method of claim 8, wherein the exception is returned to all processes involved in the deadlock.

11. The method of claim 1, wherein the examination of processes queued on monitors to determine whether there is a deadlock situation includes processes having a conditional wait on a monitor.

12. The method of claim 1, wherein the examination of processes queued on monitors includes monitors on one or more remote machines.

13. A computing system supporting multiple processes, and including:

a set of monitors for controlling access to resources of the computer system, whereby a process has to enter a monitor in order to access a corresponding resource, and has to join an entry queue for the monitor if the resource is currently owned by another process;

means responsive to a predetermined condition for examining processes queued on monitors to determine whether there is a deadlock situation; and

means responsive to such deadlock being found for returning information about the identity of the processes and monitors involved in the deadlock.

14. The system of claim 13, wherein said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process.

15. The system of claim 14, wherein the processes to be examined are determined iteratively based on those processes queued on the requested monitor, and those

processes queued on monitors owned by processes already determined to be examined.

16. The system of claim 14, wherein the system maintains a global table of contended monitors, a monitor being contended if there is a process in its entry queue, the table identifying processes which own or are queued on contended monitors.

17. The system of claim 13, wherein said predetermined condition comprises the detection that a set of processes have not progressed over a certain interval.

18. The system of claim 17, wherein said detection is performed by a process that periodically examines the program counter of other processes on the system.

19. The system of claim 13, wherein an error is returned if deadlock is found, said error including said information about the identity of the processes and monitors involved in the deadlock.

20. The system of claim 13, wherein an exception is returned if deadlock is found, said exception including said information about the identity of the processes and monitors involved in the deadlock.

21. The system of claim 20, wherein said predetermined condition comprises a process requesting to enter a

monitor that is currently owned by another process, and said exception is returned to the requesting process.

22. The system of claim 20, wherein the exception is returned to all processes involved in the deadlock.

23. The system of claim 13, wherein the examination of processes queued on monitors to determine whether there is a deadlock situation includes processes having a conditional wait on a monitor.

24. The system of claim 13, wherein the examination of processes queued on monitors includes monitors on one or more remote machines.

25. A computer program product comprising program instructions encoded in machine readable form on a medium, said instructions when loaded into a computer system causing the system to perform the steps of:

providing a set of monitors for controlling access to resources of the computer system, whereby a process has to enter a monitor in order to access a corresponding resource, and has to join an entry queue for the monitor if the resource is currently owned by another process;

responsive to a predetermined condition, examining processes queued on monitors to determine whether there is a deadlock situation; and

if such a deadlock is found, returning information about the identity of the processes and monitors involved in the deadlock.

26. The computer program product of claim 25, wherein said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process.

27. The computer program product of claim 26, wherein the processes involved in said examining step are determined iteratively based on those processes queued on the requested monitor, and those processes queued on monitors owned by processes already determined to be involved in said examining step.

28. The computer program product of claim 26, wherein the system maintains a global table of contended monitors, a monitor being contended if there is a process in its entry queue, the table identifying processes which own or are queued on contended monitors.

29. The computer program product of claim 25, wherein said predetermined condition comprises the detection that a set of processes have not progressed over a certain interval.

30. The computer program product of claim 29, wherein said detection is performed by a process that periodically examines the program counter of other processes on the system.

31. The computer program product of claim 25, wherein an error is returned if deadlock is found, said error including said information about the identity of the processes and monitors involved in the deadlock.

32. The computer program product of claim 25, wherein an exception is returned if deadlock is found, said exception including said information about the identity of the processes and monitors involved in the deadlock.

33. The computer program product of claim 32, wherein said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process, and said exception is returned to the requesting process.

34. The computer program product of claim 32, wherein the exception is returned to all processes involved in the deadlock.

35. The computer program product of claim 25, wherein the examination of processes queued on monitors to determine whether there is a deadlock situation includes processes having a conditional wait on a monitor.

36. The computer program product of claim 25, wherein the examination of processes queued on monitors includes monitors on one or more remote machines.